A note on stochastic subgradient descent for persistence-based functionals: convergence and practical aspects

Mathieu Carriere, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hariprasad Kannan

A note on stochastic subgradient descent for persistence-based functionals: convergence and practical aspects

Mathieu Carrière* Frédéric Chazal† Marc Glisse†
Yuichi Ike‡ Hariprasad Kannan

October 19, 2020

Abstract

Solving optimization tasks based on functions and losses with a topological flavor is a very active and growing field of research in Topological Data Analysis, with plenty of applications in non-convex optimization, statistics and machine learning. All of these methods rely on the fact that most of the topological constructions are actually stratifiable and differentiable almost everywhere. However, the corresponding gradient and associated code is always anchored to a specific application and/or topological construction, and do not come with theoretical guarantees.

In this article, we study the differentiability of a general functional associated with the most common topological construction, that is, the persistence map, and we prove a convergence result of stochastic subgradient descent for such a functional. This result encompasses all the constructions and applications for topological optimization in the literature, and comes with code that is easy to handle and mix with other non-topological constraints, and that can be used to reproduce the experiments described in the literature.

1 Introduction

Persistent homology is a central tool in Topological Data Analysis (TDA) that allows to efficiently infer relevant topological features of complex data in a descriptor called persistence diagram. It found many applications in Machine Learning (ML) where it initially played the role of a feature engineering tool through the direct use of persistence diagrams or through dedicated ML architectures to handle them—see, e.g. [HKNU17, Ume17, CCI+20, DUC20, KKZ+20].

*DataShape, Inria Sophia-Antipolis, Biot, France
†DataShape, Inria Saclay, Palaiseau, France
‡Fujitsu Laboratories, AI Lab, Tokyo, Japan

For the last few years, a growing number of works have been using persistence theory in different ways in order to, for instance, better understand, design and improve neural network architectures—see, e.g. [RTB$^+$18, MHRB19, CG20, GC19]—or design regularization and loss functions incorporating topological terms and penalties for various ML tasks—see, e.g. [CNBW19, HKND19, CBO$^+$20]. These new use cases of persistence generally involve minimizing functions that depend on persistence diagrams. Such functions are in general highly non-convex and not differentiable, and thus their theoretical and practical minimization can be difficult. In some specific cases, persistence-based functions can be designed to be differentiable and/or some effort have to be made to compute their gradient, so that standard gradient descent techniques can be used to minimize them—see e.g. [WLSC20, PSO18, BGGSSG20]. In the general case, recent attempts have been made to better understand their differential structures [LOT19], and, building on the powerful tools provided by software libraries such as PyTorch or TensorFlow, practical methods that allow to encode and optimize a large family of persistence-based functions have been proposed and experimented [GNDS20, SWB20]. However, in all these cases, the algorithms used to minimize these functions do not come with theoretical guarantees of convergence to a global or local minimum.

The aim of this note is to show that the algorithm to compute persistence diagrams has a simple combinatorial structure that allows us to easily implement and use stochastic subgradient descent in a general framework. This gives convergence guarantees for a wide class of functions that includes almost all persistence-based functions in the literature. More precisely,

(1) we show that the map converting a filtration over a given simplicial complex into a persistence diagram is semi-algebraic with very easy to compute (sub-)differential;

(2) building on the recent work [DDKL20], we show that under mild assumptions, stochastic subgradient descent algorithms applied to functions of persistence converge almost surely to a critical point of the function;

(3) we provide a simple corresponding Python implementation for minimizing functions of persistence that is publicly available at `https://github.com/MathieuCarriere/difftda` (and that is going to be part of the Gudhi library [MBGY14] soon), and we illustrate it with a few examples from the literature.

The article is organized as follows. In Section 2, we introduce notations and recall basic facts about filtrations and the persistence computation algorithm. In Section 3, we study the differentiability of persistence-based maps defined in o-minimal structures, and we also provide a few examples showing that almost all persistence-based maps encountered in practice belong to such structures. In Section 4, we consider the minimization problem of a persistence-based functional. In Section 5, we provide simple implementation for persistence-based function minimization and illustrate it on a few examples. Finally, we conclude in Section 6.

2 Filtrations and persistence diagrams

In this section, we show that, given a finite simplicial complex K, the map that associates to a filtration of K its persistence diagram can be seen as a simple permutation of the coordinates of a vector containing the filtration values.

2.1 Simplicial complexes and filtrations

Recall that given a set V, a *simplicial complex* K is a collection of finite subsets of V that satisfies

(1) $\{v\} \in K$ for any $v \in V$,

(2) if $\sigma \in K$ and $\tau \subseteq \sigma$ then $\tau \in K$.

An element $\sigma \in K$ with $|\sigma| = k + 1$ is called a k-simplex.

Given a simplicial complex K and a subset R of $\mathbb{R}$, a *filtration* of K is an increasing sequence $(K_r)_{r\in R}$ of subcomplexes of K with respect to the inclusion—i.e., $K_r \subseteq K_s$ for any $r < s$—such that $\bigcup_{r\in R} K_r = K$.

To each simplex $\sigma \in K$, one can associate its *filtering index* $\Phi_\sigma = \inf\{r \in R : \sigma \in K_r\}$. Thus, when K is finite, a filtration of K can be conveniently represented as a filtering function $\Phi\colon K \to \mathbb{R}$. Equivalently, it can be represented as a $|K|$-dimensional vector $\Phi = (\Phi_\sigma)_{\sigma\in K}$ in $\mathbb{R}^{|K|}$ whose coordinates are the indices of the simplices of K and that satisfies the following condition: if $\sigma, \tau \in K$ and $\tau \subseteq \sigma$, then $\Phi_\tau \leq \Phi_\sigma$. As a consequence, if the vectorized filtration Φ depends on a parameter, the corresponding family of filtrations can be represented as a map from the space of parameters to $\mathbb{R}^{|K|}$ in the following way.

Definition 2.1. Let K be a finite simplicial complex and A a set. A map $\Phi\colon A \to \mathbb{R}^{|K|}$ is said to be a *parametrized family of filtrations* if for any $x \in A$ and $\sigma, \tau \in K$ with $\tau \subseteq \sigma$, one has $\Phi_\tau(x) \leq \Phi_\sigma(x)$.

2.2 Computation of persistence diagrams from filtrations

In this section, we briefly recall how the computation of the persistence diagram of a filtered simplicial complex decomposes into: (1) a purely combinatorial part only relying on the order on the simplices induced by the filtration, and (2) a part relying on the filtration values. For a complete introduction to persistent homology and its computation, we refer the reader to standard articles and textbooks, such as [ZC05, EH10, BCY18].

Let K be a finite simplicial complex endowed with a filtration and corresponding filtering function $\Phi\colon K \to \mathbb{R}$. We regard Φ as a vector Φ in $\mathbb{R}^{|K|}$, where $|K|$ is the number of non-empty simplices of K. The computation of the persistence diagram can be split into two steps.

First part: combinatorial part (persistence pairs). The filtering function Φ induces a preorder on the elements of K as follows: $\tau \preceq \sigma$ if $\Phi_\tau \leq \Phi_\sigma$. This preorder can be refined into a total order in some arbitrary way (e.g., by indexing the simplices, and using the lexicographic order to break ties). Note that the choice of this refinement does not change the output of the persistence computation, as long as it is deterministic and consistent. The basic algorithm to compute persistence then iterates over the ordered set of simplices $\sigma_1 \preceq \cdots \preceq \sigma_{|K|}$ in the following way (see [BCY18, Section 11.5.2] for a complete description of the algorithm).

Algorithm 1 Persistence pairs computation (sketch of)

Input: Ordered sequence of simplices $\sigma_1 \preceq \cdots \preceq \sigma_{|K|}$.
$K_0 = \emptyset$
$\text{Pairs}_0 = \text{Pairs}_1 = \cdots = \text{Pairs}_{d-1} = \emptyset$
for $j = 1$ to $|K|$ **do**
  $k = \dim \sigma_j$
  $K_j = K_{j-1} \cup \sigma_j$
  if σ_j does not create a new k-dimensional homology class in K_j **then**
    a $(k-1)$-dimensional homology class created in $K_{l(j)}$ for some $l(j) < j$ becomes homologous to 0 in K_j.
    $\text{Pairs}_{k-1} \leftarrow \text{Pairs}_{k-1} \cup \{(\sigma_{l(j)}, \sigma_j)\}$;
Output: Persistence pairs in each dimension $\text{Pairs}_0, \text{Pairs}_1, \ldots, \text{Pairs}_{d-1}$;

Note that for each dimension k, some k-dimensional simplices may remain unpaired at the end of the algorithm; their number is equal to the k-dimensional Betti number of K.

Second part: associate filtration values. The persistence diagram of the filter function Φ is now obtained by associating to each persistent pair $(\sigma_{l(j)}, \sigma_j)$ the point $(\Phi_{\sigma_{l(j)}}, \Phi_{\sigma_j})$. Moreover, to each unpaired simplex σ_l is associated the point $(\Phi_{\sigma_l}, +\infty)$.

If p is the number of persistence pairs and q is the number of unpaired simplices, then $|K| = 2p + q$ and the persistence diagram $D(\Phi)$ of the filtration Φ of K is made of p points in $\mathbb{R}^2$ (counted with multiplicity) and q points (also counted with multiplicity) with infinite second coordinate. Choosing an order (e.g., the lexicographical order) on $\mathbb{R} \times (\mathbb{R} \cup \{+\infty\})$, the persistence diagram $D = D(\Phi)$ can be represented as a vector in $\mathbb{R}^{|K|}$ and the output of the persistence algorithm can be simply seen as a permutation of the coordinates of the input vector Φ. Moreover, this permutation only depends on the order on the simplices of K induced by Φ.

Definition 2.2. The subset of points of a persistence diagram D with finite coordinates is called the *regular part* of D and is denoted by D_{reg}. The subset of points of D with infinite second coordinate is called the *essential part* of D and is denoted by D_{ess}.

With the notations defined above, D_{reg} and D_{ess} can be represented as vectors in $\mathbb{R}^{2p}$ and $\mathbb{R}^q$, respectively.

Remark 2.3. Note that the above construction is usually done dimension by dimension, leading to a single persistence diagram for each dimension in homology. Indeed, in order to obtain a persistence diagram in dimension k, it suffices to restrict to the subset of simplices of dimension k and $k+1$. Without loss of generality, and to avoid unnecessary heavy notations, in the remaining of this article, we consider the whole persistence diagram, made of the union of the persistence diagrams in all dimensions k.

3 Differentiability of persistence maps defined in o-minimal structures

In most cases encountered in practice, the parametrized families of filtrations that are used are definable in o-minimal structures. Moreover, given a finite simplicial complex K, the persistence map that associates its persistence diagram to each filtration on K is semi-algebraic. As a consequence, its composition with a parametrized family of filtrations is definable and benefits from interesting differentiability properties.

3.1 Background on o-minimal geometry

In this section, we briefly recall some elements of o-minimal geometry, which are needed in the next sections of this article. For a more detailed introduction, we refer to standard textbooks, such as [Cos00].

Definition 3.1 (o-minimal structure)**.** An *o-minimal structure* on the field of real numbers $\mathbb{R}$ is a collection $(S_n)_{n\in\mathbb{N}}$, where each S_n is a set of subsets of $\mathbb{R}^n$ such that:

(1) S_1 is exactly the collection of the finite unions of points and intervals;

(2) all algebraic subsets of $\mathbb{R}^n$ are in S_n;

(3) S_n is a Boolean subalgebra of $\mathbb{R}^n$ for any $n \in \mathbb{N}$;

(4) if $A \in S_n$ and $B \in S_m$, then $A \times B \in S_{n+m}$;

(5) if $\pi\colon \mathbb{R}^{n+1} \to \mathbb{R}^n$ is the linear projection onto the first n coordinates and $A \in S_{n+1}$, then $\pi(A) \in S_n$.

An element $A \in S_n$ for some $n \in \mathbb{N}$ is called a *definable set* in the o-minimal structure. For a definable set $A \subseteq \mathbb{R}^n$, a map $f\colon A \to \mathbb{R}^m$ is said to be *definable* if its graph is a definable set in $\mathbb{R}^{n+m}$.

Definable sets are stable under various geometric operations. The complement, closure and interior of a definable set are definable sets. The finite unions and intersections of definable sets are definable. The image of a definable set

by a definable map is itself definable. Sums and products of definable functions as well as compositions of definable functions are definable—see [Cos00, Section 1.3]. In particular, the max and min of finite sets of real-valued definable functions are also definable. An important property of definable sets and definable maps is that they admit a finite Whitney stratification. This implies that (1) any definable set $A \subseteq \mathbb{R}^n$ can be decomposed into a finite disjoint union of smooth submanifolds of $\mathbb{R}^n$ and (2) for any definable map $\Phi\colon A \to \mathbb{R}^m$, A can also be decomposed into a finite union of smooth manifolds such that the restriction of Φ on each of these manifolds is a smooth function.

The simplest example of o-minimal structures is given by the family of semi-algebraic subsets of the spaces $\mathbb{R}^n$ ($n \in \mathbb{N}$). Although most of the parametrized families of filtrations encountered in practice are semi-algebraic, the o-minimal framework actually allows one to consider families that mix exponential functions with semi-algebraic functions [Wil96].

3.2 Persistence diagrams of definable parametrized families of filtrations

Let K be a finite simplicial complex and $\Phi\colon A \to \mathbb{R}^{|K|}$ be a definable (in a given o-minimal structure) parametrized family of filtrations. If for any $x, x' \in A$, the preorders induced by $\Phi(x)$ and $\Phi(x')$ on the simplices of K are the same, i.e., for any $\sigma_1, \sigma_2 \in K$, $\Phi_{\sigma_1}(x) \leq \Phi_{\sigma_2}(x)$ if and only if $\Phi_{\sigma_1}(x') \leq \Phi_{\sigma_2}(x')$, then the pairs of simplices $(\sigma_{i_1}, \sigma_{j_1}), \ldots, (\sigma_{i_p}, \sigma_{j_p})$, and the unpaired simplices $\sigma_{i_{p+1}}, \ldots, \sigma_{i_{p+q}}$ that are computed by the persistence algorithm (see Algorithm 1) are independent of x. As a consequence, the persistence diagram of $\Phi(x)$ is

$$D = D(\Phi(x)) = \bigcup_{k=1}^{p} (\Phi_{\sigma_{i_k}}(x), \Phi_{\sigma_{j_k}}(x)) \cup \bigcup_{k=1}^{q} (\Phi_{\sigma_{i_{p+k}}}(x), +\infty), \tag{3.1}$$

where $|K| = 2p + q$.

Given a total order on $\mathbb{R} \times (\mathbb{R} \cup \{+\infty\})$, the points of any finite multi-set $D \subseteq \mathbb{R} \times (\mathbb{R} \cup \{+\infty\})$ with p points in $\mathbb{R}^2$ and q points in $\mathbb{R} \times \{+\infty\}$ can be ordered in non-decreasing order, and D can be represented as a vector in $\mathbb{R}^{2p+q}$. In the following, we assume that such an order is given and fixed. As a consequence, denoting by Filt_K the set of vectors in $\mathbb{R}^{|K|}$ defining a filtration on K, the persistence map $\mathrm{Pers}\colon \mathrm{Filt}_K \to \mathbb{R}^{|K|}$ that assigns to each filtration of K its persistence diagram consists in a permutation of the coordinates of $\mathbb{R}^{|K|}$. This permutation is constant on the set of filtrations that define the same preorder on the simplices of K. This leads to the following statement.

Proposition 3.2. *Given a finite simplicial complex K, the map* $\mathrm{Pers}\colon \mathrm{Filt}_K \subseteq \mathbb{R}^{|K|} \to \mathbb{R}^{|K|}$ *is semi-algebraic, and thus definable in any o-minimal structure. Moreover, there exists a semi-algebraic partition of* Filt_K *such that the restriction of* Pers *to each element of this partition is a Lipschitz map.*

Proof. As K is finite, the number of preorders on the simplices of K is finite. Let $\preceq$ be a preorder on simplices of K, induced by some equalities and inequalities

between the coordinates of $\mathbb{R}^{|K|}$. Then, the set of filtrations $F \in \mathrm{Filt}_K$ such that F gives rise to a preorder equal to $\preceq$ is a semi-algebraic set. It follows that Filt_K is a semi-algebraic set that admit a semi-algebraic partition such that the restriction of the persistence map Pers to each component is a constant permutation.

Now, from the stability theorem for persistence [CdSGO16], the persistence modules induced by two filtrations $F_1, F_2 \in \mathrm{Filt}_K$ are ϵ-interleaved where ϵ is the sup norm of the vector $F_2 - F_1$. As a consequence, the restriction of Pers to each component of the above mentioned semi-algebraic partition of Filt_K is clearly 1-Lipschitz with respect to the sup norm in $\mathbb{R}^{|K|}$. □

Since there exists a finite semi-algebraic partition of Filt_K on which Pers is a locally constant permutation, the subdifferential of Pers is well-defined and obvious to compute: each coordinate in the output (i.e., the persistence diagram) is a copy of a coordinate in the input (i.e., the filtration values of the simplices). This implies that every partial derivative is either 1 or 0. The output can be seen as a reindexing of the input, and this is indeed how we implement it in our code, so that automatic differentiation frameworks (PyTorch, TensorFlow, etc.) can process the function Pers directly and do not need explicit gradient formulas—see Section 5.

Corollary 3.3. *Let K be a finite simplicial complex and $\Phi\colon A \to \mathbb{R}^{|K|}$ be a definable (in a given o-minimal structure) parametrized family of filtrations. The map $\mathrm{Pers} \circ \Phi\colon A \to \mathbb{R}^{|K|}$ is definable.*

Note that according to the remark following Proposition 3.2, if Φ is differentiable, the subdifferential of $\mathrm{Pers} \circ \Phi$ can be easily computed in terms of the partial derivatives of Φ using, for example, Equation (3.1).

It also follows from standard finiteness and stratifiability properties of definable sets and maps that $\mathrm{Pers} \circ \Phi$ is differentiable almost everywhere. More precisely, we have the following result.

Proposition 3.4. *Let K be a finite simplicial complex and $\Phi\colon A \to \mathbb{R}^{|K|}$ a definable parametrized family of filtrations. Then there exists a finite definable partition of A, $A = S \sqcup O_1 \sqcup \cdots \sqcup O_k$ such that*

(1) $\dim S < \dim A := m$;

(2) *for any $i = 1, \ldots, k$, O_i is a definable manifold of dimension m and $\mathrm{Pers} \circ \Phi\colon O_i \to \mathcal{D}$ is differentiable.*

3.3 Examples of definable families of filtrations

Below we provide a few examples of common families of filtrations that turn out to be semi-algebraic or definable in a more general o-minimal structure.

Example 3.5 (Vietoris-Rips filtrations)**.** The family of Vietoris-Rips filtrations built on top of sets of n points $x_1, \ldots, x_n \in \mathbb{R}^d$ is the semi-algebraic parametrized

family of filtrations

$$\Phi\colon A = (\mathbb{R}^d)^n \to \mathbb{R}^{|\Delta_n|} = \mathbb{R}^{2^n-1},$$

where Δ_n is the simplicial complex made of all the faces of the $(n-1)$-dimensional simplex, defined, for any $(x_1, \ldots, x_n) \in A$ and any simplex σ spanned by a subset $J \subseteq \{1, \ldots, n\}$, by

$$\Phi_\sigma(x) = \max_{i,j \in J} \|x_i - x_j\|.$$

One can easily check that the permutation induced by Pers is constant on the connected components of the complement of the union of the subspaces $S_{i,j,k,l} = \{(x_1, \ldots, x_n) : \|x_i - x_j\| = \|x_k - x_l\|\}$ over all the 4-tuples (i, j, k, l) such that at least 3 of the 4 indices i, j, k, l are distinct.

This example naturally extends to the case of Vietoris-Rips-like filtrations in the following way. Let $A \subset \mathcal{M}_n(\mathbb{R})$ be the set of $n \times n$ symmetric matrices with non-negative entries and 0 on the diagonal[1]. This is a semi-algebraic subset of the space of n-by-n matrices $\mathcal{M}_n(\mathbb{R}) \simeq \mathbb{R}^{n^2}$, of dimension $m = (n-1)(n-2)/2$. The map $\Phi\colon A \to \mathbb{R}^{|\Delta_n|} = \mathbb{R}^{2^n}$ defined by

$$\Phi_\sigma(M) = \max_{i,j \in J} m_{i,j}$$

for any $M = (m_{i,j})_{1 \leq i,j \leq n} \in A$ and any simplex σ spanned by a subset $J \subseteq \{1, \ldots, n\}$, is a semi-algebraic family of filtrations. Note that the set S of Proposition 3.4 can be chosen to be the $(m-1)$-dimensional semi-algebraic set of matrices with at least 2 entries that are equal.

Example 3.6 (Weighted Rips-filtrations)**.** Weighted Rips filtrations are a generalization of Vietoris-Rips filtrations where weights are assigned to the vertices of the simplicial complex. Given a set of n points $x_1, \ldots, x_n \in \mathbb{R}^d$ and a function $f\colon \mathbb{R}^d \to \mathbb{R}$, the family of weighted Rips filtrations $\Phi\colon A = (\mathbb{R}^d)^n \to \mathbb{R}^{|\Delta_n|} = \mathbb{R}^{2^n}$ associated to f is defined for any simplex σ spanned by $J \subseteq \{1, \ldots, n\}$ by

- $\Phi_\sigma(x) = 2f(x_j)$ if $\sigma = [j]$;
- $\Phi_\sigma(x) = \max(2f(x_i), 2f(x_j), \|x_i - x_j\| + f(x_i) + f(x_j)$, if $\sigma = [i, j]$, $i \neq j$;
- $\Phi_\sigma(x) = \max(\Phi_{[i,j]}(x), i, j \in J)$ if $|J| \geq 3$.

As Euclidean distances between points are semi-algebraic, as well as the max function, it is clear from this definition that a family of weighted Rips filtrations is definable as soon as the weight function f is definable.

This example easily extends to the case where the weight function depends on the set of points $X_n = (x_1, \ldots, x_n)$: the weight at vertex x is defined by $f(X_n, x)$ with $f\colon (\mathbb{R}^d)^n \times \mathbb{R}^d \to \mathbb{R}$. Again, the resulting family of filtrations is definable as soon as f is definable. A particular example of such a family is

[1] Note that this set contains the set of matrices of pairwise distances between sets of n points in metric spaces.

given by the so-called DTM filtration [ACG+20], where $f(X_n, x)$ is the average distance from x to its k-nearest neighbors in X_n. In this case, f is semi-algebraic, and the family of DTM filtrations is semi-algebraic.

The o-minimal framework also allows to consider weight functions involving exponential functions, such as, for instance, kernel-based density estimates with Gaussian kernel. This is a consequence of the fundamental result of [Wil96] in o-minimal geometry.

Example 3.7 (Sublevel sets filtrations)**.** Let K be a finite simplicial complex with n vertices $v_1, \ldots, v_n$. Any real-valued function f defined on the vertices of K can be represented as a vector $(f(v_1), \ldots, f(v_n)) \in \mathbb{R}^n$. The family of sublevel sets filtrations $\Phi\colon A = \mathbb{R}^n \to \mathbb{R}^{|K|}$ of functions on the vertices of K is defined by

$$\Phi_\sigma(f) = \max_{i \in J} f_i$$

for any $f = (f_1, \ldots, f_n) \in A$ and any simplex σ spanned by a subset $J \subseteq \{1, \ldots, n\}$. This filtration is also known as the *lower-star filtration* of f. The function Φ is obviously semi-algebraic, and for Proposition 3.4 it is sufficient to choose $S = \bigcup_{1 \leq i < j \leq n} \{f = (f_1, \ldots, f_n) \in A : f_i = f_j\}$.

4 Minimization of functions of persistence

Using the same notation as in the previous section, recall that the space of persistence diagrams associated to a filtration of K is identified with $\mathbb{R}^{|K|} = (\mathbb{R}^2)^p \times \mathbb{R}^q$, where each point in the p copies of $\mathbb{R}^2$ is a point with finite coordinates in the persistence diagram and each coordinate in $\mathbb{R}^q$ is the x-coordinate of a point with infinite persistence.

Definition 4.1. A function $E\colon \mathbb{R}^{|K|} = (\mathbb{R}^2)^p \times \mathbb{R}^q \to \mathbb{R}$ is called a *function of persistence* if it is invariant to permutations of the points of the persistence diagram: for any $(p_1, \ldots, p_p, b_1, \ldots, b_q) \in (\mathbb{R}^2)^p \times \mathbb{R}^q$ and any permutations α, β of the sets $\{1, \ldots, p\}$ and $\{1, \ldots, q\}$, respectively, one has

$$E(p_{\alpha(1)}, \ldots, p_{\alpha(p)}, b_{\beta(1)}, \ldots, b_{\beta(q)}) = E(p_1, \ldots, p_p, b_1, \ldots, b_q).$$

Remark 4.2. It follows from the persistence stability theorem [CdSGO16] that if a function of persistence E is locally Lipschitz, then the composition $E \circ \mathrm{Pers}$ is also locally Lipschitz.

If $E\colon \mathbb{R}^{2p+q} = \mathbb{R}^{|K|} \to \mathbb{R}$ is a definable function of persistence, then for any definable parametrized family of filtrations $\Phi\colon A \subseteq \mathbb{R}^d \to \mathbb{R}^{|K|}$, the composition $\mathcal{L} = E \circ \mathrm{Pers} \circ \Phi\colon A \to \mathbb{R}$ is also definable. As a consequence, it has a well-defined subgradient. Building on a recent result from [DDKL20], the aim of this section is to show that stochastic subgradient descent applied to $\mathcal{L}$ converges almost surely to a critical point under mild conditions on $\mathcal{L}$. We show that these conditions are satisfied for a wide variety of examples commonly encountered in Topological Data Analysis.

4.1 Stochastic gradient descent

To minimize $\mathcal{L}$, we consider the differential inclusion

$$\frac{dz}{dt} \in -\partial\mathcal{L}(z(t)) \quad \text{for almost every } t,$$

whose solutions $z(t)$ are the trajectories of the gradient of $\mathcal{L}$. They can be approximated by the standard stochastic subgradient algorithm given by the iterations of

$$x_{k+1} = x_k + \alpha_k(y_k + \zeta_k),\ y_k \in \partial\mathcal{L}(x_k), \tag{4.1}$$

where $\partial\mathcal{L}$ denotes the subgradient of $\mathcal{L}$, the sequence $(\alpha_k)_k$ is the learning rate, and $(\zeta_k)_k$ is a sequence of random variables. In [DDKL20], the authors prove that under mild technical conditions on these two sequences, the stochastic subgradient algorithm converges almost surely to a critical point of $\mathcal{L}$ as soon as $\mathcal{L}$ is locally Lipschitz.

More precisely, consider the following assumptions (which correspond to Assumption C in [DDKL20]):

(1) for any k, $\alpha_k \geq 0$, $\sum_{k=1}^{\infty} \alpha_k = +\infty$, and $\sum_{k=1}^{\infty} \alpha_k^2 < +\infty$;

(2) $\sup_k \|x_k\| < +\infty$, almost surely;

(3) denoting by $\mathcal{F}_k$ the increasing sequence of σ-algebras $\mathcal{F}_k = \sigma(x_j, y_j, \zeta_j, j \leq k)$, there exists a function $p\colon \mathbb{R}^d \to \mathbb{R}$ which is bounded on bounded sets such that almost surely, for any k,

$$\mathbb{E}[\zeta_k|\mathcal{F}_k] = 0 \ \text{ and } \ \mathbb{E}[\|\zeta_k\|^2|\mathcal{F}_k] < p(x_k).$$

These assumptions are standard and not very restrictive. Assumption (1) depends on the choice of the learning rate by the user and is easily satisfied, e.g., taking $\alpha_k = 1/k$. Assumption (2) is usually easy to check for most of the functions $\mathcal{L}$ encountered in practice (see below). Assumption (3) is a standard condition, which states that, conditioned upon the past, the variables ζ_k have zero mean and controlled moments; e.g., this can be achieved by taking a sequence of independent and centered variables with bounded variance that are also independent of the x_k's and y_k's.

Under these assumptions (1), (2), and (3), the following result is an immediate consequence of Corollary 5.9 in [DDKL20].

Theorem 4.3. *Let K be a finite simplicial complex, $A \subseteq \mathbb{R}^d$, and $\Phi\colon A \to \mathbb{R}^{|K|}$ a parametrized family of filtrations of K that is definable in an o-minimal structure. Let $E\colon \mathbb{R}^{|K|} \to \mathbb{R}$ be a definable real-valued function defined on the space of persistence diagrams such that $\mathcal{L} = E \circ \mathrm{Pers} \circ \Phi$ is locally Lipschitz. Then, under the above assumptions (1), (2), and (3), almost surely the limit points of the sequence $(x_k)_k$ obtained from the iterations of Equation* (4.1) *are critical points of $\mathcal{L}$ and the sequence $(\mathcal{L}(x_k))_k$ converges.*

The above theorem provides explicit conditions ensuring the convergence of stochastic gradient descent for functions of persistence. The main criterion to be checked is the local Lipschitz condition for $\mathcal{L}$. From Remark 4.2, it is enough to check that Φ and E are Lipschitz. Regarding Φ, while it is obvious for Rips, Čech, DTM, etc. filtrations on finite point clouds, it is not as naturally satisfied by the alpha-complex filtration, where 3 points with diameter 1 can create an edge and a triangle with an arbitrarily large filtration value. However, such large filtration values only ever participate in persistence intervals of length 0 (points on the diagonal of the diagram): indeed, the alpha-complex filtration and the Čech filtration have the same persistence diagram, if we ignore the diagonal. Filtering out 0-length intervals from the alpha-complex thus gives a locally Lipschitz function and can be seen as a technical detail of computing the Čech filtration.

4.2 Examples of locally Lipschitz functions of persistence

Now, regarding the function E, many functions of persistence considered in the literature are Lipschitz or locally Lipschitz, as illustrated by the following, non-exhaustive, list of examples.

Example 4.4 (Total persistence)**.** The total persistence function E is the sum of the distances of each point of a persistence diagram with finite coordinates to the diagonal: given a persistence diagram represented as a vector in $\mathbb{R}^{2p+q}$, $D = ((b_1, d_1), \ldots, (b_p, d_p), e_1, \ldots, e_q)$,

$$E(D) = \sum_{i=1}^{p} |d_i - b_i|.$$

Thus, E is obviously semi-algebraic, and thus definable in any o-minimal structure. It is also Lipschitz.

Example 4.5 (Wasserstein and bottleneck distance)**.** Given a persistence diagram D^0, the bottleneck distance between the regular part of a diagram D and the regular part of D^* (see Definition 2.2) is given by

$$E(D) = d_B(D_{\text{reg}}, D^*_{\text{reg}}) = \min_{\text{matching } m} \max_{(p,p^*)\in m} ||p - p^*||_\infty,$$

where, denoting $\Delta = \{(x, x) : x \in \mathbb{R}\}$ the diagonal in $\mathbb{R}^2$, a matching m is a subset of $(D_{\text{reg}} \cup \Delta) \times (D^*_{\text{reg}} \cup \Delta)$ such that every point of $D_{\text{reg}} \setminus \Delta$ and $D^*_{\text{reg}} \setminus \Delta$, appears exactly once in m. One can easily check that the map E is semi-algebraic, and thus definable in any o-minimal structure. It is also Lipschitz. This property also extends to the case where the bottleneck distance is replaced by the so-called Wasserstein distance W_p with $p \in \mathbb{N}$ [CSEHM10]. Optimization of these functionals E and other functionals of bottleneck and Wasserstein distances have been used, for example, in shape matching in [PSO18], see also the example on 3D shape in Section 5.

Example 4.6 (Persistence landscapes [Bub15])**.** To any given a point $p = (x, y) \in \mathbb{R}^2$ with $x = \frac{b+d}{2}$ and $y = \frac{d-b}{2}$, associate the function $\Lambda_p \colon \mathbb{R} \to \mathbb{R}$ defined by

$$\Lambda_p(t) = \begin{cases} t - x + y & t \in [x - y, x] \\ x + y - t & t \in (x, x + y] \\ 0 & \text{otherwise} \end{cases} = \begin{cases} t - b & t \in [b, \frac{b+d}{2}] \\ d - t & t \in (\frac{b+d}{2}, d] \\ 0 & \text{otherwise.} \end{cases}$$

Given a persistence diagram D, the persistence landscape of D is a summary of the arrangement of the graphs of the functions Λ_p, $p \in D$:

$$\lambda_D(k, t) = \underset{p \in \mathrm{dgm}}{k\text{-max}}\, \Lambda_p(t), \quad t \in [0, T], k \in \mathbb{Z}^+,$$

where k-max is the kth largest value in the set, or 0 when the set contains less than k points. An interesting property of persistence landscapes is that the map from the space of persistence diagrams endowed with the bottleneck distance to the vector space of real-valued functions on $\mathbb{Z}^+ \times \mathbb{R}$ endowed with the sup norm is 1-Lipschitz—see [Bub15] or [CFL$^+$15, Lemma 1].

Given a positive integer k, a finite set $\{t_1, \ldots, t_n\} \subset \mathbb{R}$, and a finite simplicial complex K, the map that associates the vector $(\lambda_D(k, t_1), \ldots, \lambda_D(k, t_n))$ to each persistence diagram D of a filtration of K is clearly semi-algebraic. As a consequence, Theorem 4.3 applies to any locally Lipschitz definable function $E \colon \mathbb{R}^n \to \mathbb{R}$ composed with the map $D \mapsto (\lambda_D(k, t_1), \ldots, \lambda_D(k, t_n))$. Functions of landscapes have been used, for example, to design topological layers in neural networks [KKZ$^+$20].

Example 4.7 (Persistence surfaces [AEK$^+$17])**.** Given a weight function $w \colon \mathbb{R}^2 \to \mathbb{R}^+$ and a real number $\sigma > 0$, the persistence surface associated to a persistence diagram D is the function $\rho_D \colon \mathbb{R}^2 \to \mathbb{R}$ defined by

$$\rho_D(q) = \sum_{p \in D} w(p) \exp\left(-\frac{\|q - p\|^2}{2\sigma^2}\right).$$

The definition of ρ_D only involves algebraic operations, the weight function w, and exponential functions. Thus, it follows from [Wil96] that if w is a semi-algebraic function, K is a finite simplicial complex, and $\{q_1, \ldots, q_n\}$ is a finite set of points in $\mathbb{R}^2$, then the map that associates the vector $(\rho_D(q_1), \ldots, \rho_D(q_n))$ to each persistence diagram D of a filtration of K is definable in some o-minimal structure. In [AEK$^+$17], the authors provide explicit conditions under which the map from persistence diagrams to persistence surfaces is Lipschitz.

Persistence surfaces belong to the set of *linear representations* of persistence diagrams [CD18], which includes many other ways to vectorize persistence diagrams. In [DL19], the authors give explicit conditions for such representations to be locally Lipschitz.

5 Implementation and numerical illustrations

We showed in Sections 3 and 4 that the usual stochastic gradient descent procedure of Equation (4.1) enjoys some convergence properties for persistence diagrams and persistence-based functions. This means in particular that the algorithms available in standard libraries such as TensorFlow, which implement stochastic gradient descent among other optimization methods, can be leveraged and used as is for differentiating persistence diagrams, while still ensuring convergence. The purpose of this section is to demonstrate that our code, based on Gudhi and TensorFlow, can be readily used for several different persistence optimization tasks that were presented in the literature. Our code is publicly available online[2], as well as some other code for generating the images presented below. In the following, we describe a few optimization tasks that were studied in Topological Data Analysis.

Point cloud optimization. A toy example in topological optimization is to modify the positions of the points in a point cloud so that its homology is maximized [GNDS20, GHO16]. In this paragraph, we present two numerical experiments in this vein.

In our first experiment, we start with a point cloud X sampled uniformly from the unit square $S = [0,1]^2$, and optimize the point coordinates, so that the loss $\mathcal{L}(X) = P(X) + T(X)$ is minimized, where $T(X) := -\sum_{p \in D} \|p - \pi_\Delta(p)\|_\infty^2$ is a topological penalty, D is the 1-dimensional persistence diagram associated to the Vietoris-Rips filtration of X, and π_Δ stands for the projection onto the diagonal Δ, and where $P(X) := \sum_{x \in X} d(x, S)$ is a penalty term ensuring that the point coordinates stay in the unit square. Using this loss ensures that holes are created within the point cloud so that the cardinality of the persistence diagram D is as large as possible, while constraining the points to stay in a fixed region S of the Euclidean plane. Another effect of the penalty $P(X)$ is to flatten the boundary of the created holes along the boundary of S. See the first row of Figure 5.1. It is important to note that without the penalty $P(X)$, convergence is very difficult to reach since inflating the point cloud with dilations can make the topological penalty $T(X)$ arbitrarily small.

In our second experiment, we start with a noisy sample X of the circle with three outliers and use the loss $\mathcal{L}(X) = W_2(D, D^*)^2$, where W_2 stands for the Wasserstein distance (see Example 4.5), D is the 0-dimensional persistence diagram associated to the Vietoris-Rips filtration of X, and D^* is the 0-dimensional persistence diagram associated to the Vietoris-Rips filtration of a clean (i.e., with no noise nor outliers) sample of the circle. See the second row of Figure 5.1. It is interesting to note that when one does not use extra penalties, optimizing only topological penalties can lead to funny effects: as one can see on the lower middle of Figure 5.1, the circle got disconnected, and one of the outliers created a small path out of the circle during optimization.

[2] `https://github.com/MathieuCarriere/difftda`

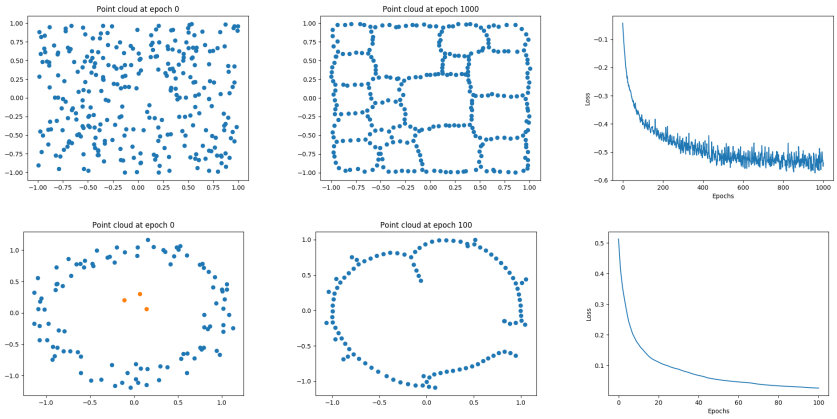

Figure 5.1: Upper row: Random point cloud initialization, before (left) and after (middle) optimization, and corresponding loss function (right). Lower row: Noisy circle initialization with outliers, before (left) and after (middle) optimization, and corresponding loss function (right).

Image processing. Another task presented in [GNDS20] is related to image processing. In this experiment, we optimize the 0-dimensional homology associated to the pixel values of a digit binary image I with noise (see Figure 5.2, upper left). Since noise can be detected as unwanted small connected components, we use the loss $\mathcal{L}(I) = P(I) + T(I)$, where $T(I) := \sum_{i=1}^{p} |d_i - b_i|$ is the total persistence penalty, $D_{\text{reg}} = \{(b_1, d_1), \ldots, (b_p, d_p)\}$ is the finite 0-dimensional persistence diagram of the cubical complex associated to I, and $P(I) := \sum_{p \in I} \min\{|p|, |1-p|\}$ is a penalty forcing the pixel values to be either 0 or 1. As can be seen from Figure 5.2, using the two penalties is essential: if only $P(I)$ is used, the noise is amplified (lower left), and if only $T(I)$ is used, the noise does not quite disappear, and paths are created between the noise and the central digit to ensure the corresponding connected components are merged right after they appear (lower middle). Note that this funny behavior which appears when penalizing topology alone is similar to what was observed in experiments where persistence was used to simplify functions [AGH$^+$09]. Using both penalties completely remove the noise (lower right) in two steps: firstly topology is optimized, and then the paths created by optimizing $T(I)$ are removed by optimizing $P(I)$. This two step process can also be observed on the loss function (upper middle) and the sequence of optimized persistence diagrams of the image (upper right).

3D shape processing. In [PSO18], persistence optimization is used for modifying functions defined on 3D shapes. More specifically, given a 3D shape S, one is interested in optimizing a function $F\colon V(S) \to \mathbb{R}$, defined on the vertices $V(S)$ of S, so that the Wasserstein distance between the persistence diagram associ-

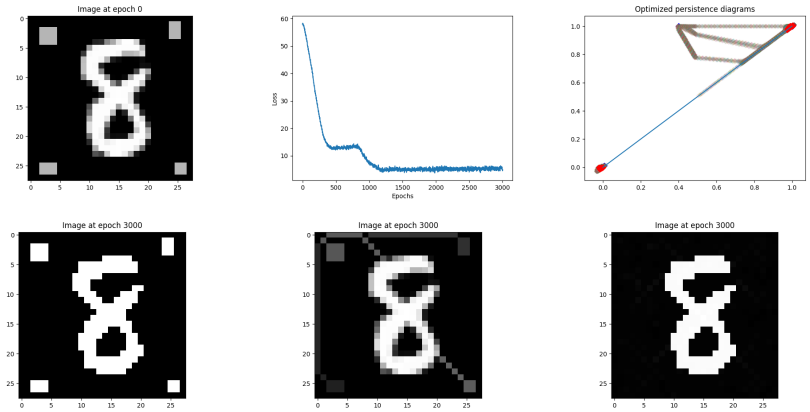

Figure 5.2: Image before (upper left) and after (lower row) optimization for various penalties. When both $T(I)$ and $P(I)$ are used, we also show the corresponding loss function (upper middle) and the sequence of persistence diagrams (upper right) with blue points being the initial persistence diagram, and red ones being the fully optimized persistence diagram.

ated to F and D^* is minimized, where D^* is a target persistence diagram (which either comes from another function $G\colon S \to \mathbb{R}$, or is defined by the user). In this experiment, we minimize the loss $\mathcal{L}(F) = T(F)$, where $T(F) := W_2(D, D^*)^2$, that is, the Wasserstein distance between the 0-dimensional persistence diagram D associated to the sublevel set of a function F—initialized with the height function, see Figure 5.3 (upper left)—of a human 3D shape, and a target persistence diagram D^* which only contains a single point with the same coordinates as the point in the persistence diagram of the height function of the shape S corresponding to the right leg. This makes the function values to force the two points in D corresponding to the hands of S to go to the diagonal, by creating paths between the hands and the hips (upper middle). It is worth noting that these path creations come from the fact that we only used a topological penalty in the loss: in [PSO18], the authors ensure smoothness of the resulting function by forcing it to be a linear combination of a first few eigenfunctions of the Laplace-Beltrami operator on the 3D shape. We also display the sequence of optimized persistence diagrams in Figure 5.3 (lower row), from which it can be observed that the optimization is piecewise-linear, which reflects the fact that the persistence map has an associated semi-algebraic partition, as per Proposition 3.2.

Linear regression. Our last experiment comes from [GNDS20], in which the authors use persistence optimization in a linear regression setting. Given some dataset $X \in \mathbb{R}^{n \times p}$ and ground-truth associated values $Y \in \mathbb{R}^n$ computed as $Y = X \cdot \beta^* + \epsilon$, where $\beta^* \in \mathbb{R}^p$ is the vector of ground-truth coefficients and ϵ is

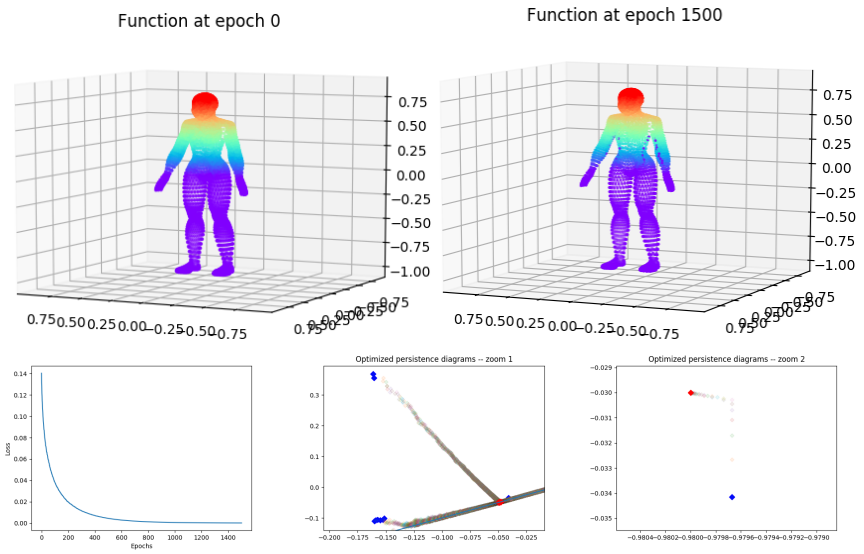

Figure 5.3: 3D shape before (upper left) and after (upper right) optimization, and corresponding loss function (lower left). Note how paths of low function values were created between the hips and the hands. We also show the sequence of persistence diagrams (lower middle and right) with blue points being the initial persistence diagram, and red ones being the fully optimized persistence diagram. Note how optimization trajectories look piecewise-linear.

some high-dimensional Gaussian noise, one can leverage some prior knowledge on the shape of β to penalize the coefficients with bad topology. In particular, when using β^* with three peaks, as in Figure 5.4 (left), we use the loss $\mathcal{L}(\beta) = P(\beta) + TV(\beta) + T(\beta)$, where $T(\beta) := \sum_{i=1}^{p} |d_i - b_i|$, $\tilde{D} = \{(b_1, d_1), \ldots, (b_p, d_p)\}$ is the 0-dimensional persistence diagram of the sublevel sets of β, minus the three most prominent points, $TV(\beta) = \sum_i |\beta_{i+1} - \beta_i|$ is the usual total variation penalty (which can also be interpreted as a topological penalty as it corresponds to the total persistence of the so-called *extended persistence* [CSEH09] of the signal), and $P(\beta) := \sum_i |x_i \cdot \beta - y_i|^2$ is the usual mean-square error (MSE). We optimized β with the MSE alone, then MSE plus total variation, then MSE plus total variation plus topological penalty, and we generated new MSE values from new randomly generated test data, see Figure 5.4 (right). It is interesting to note that using all penalties lead to the best result: using MSE alone leads to overfitting, and adding total variation smooths the coefficients a lot since β is initialized with random values. Using all penalties ends up being a right combination of minimizing the error, smoothing the signal, and getting to the right shape of β.

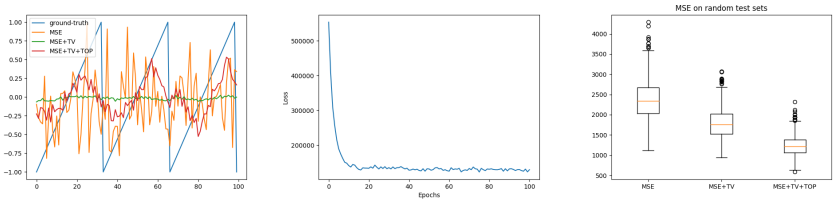

Figure 5.4: Regression coefficients after optimization for various penalties (left), and corresponding loss function when all penalties are used (middle). Generalization performance is increased by using all penalties, since the MSE on various randomly generated test sets is largely decreased (right).

6 Conclusion

In this article, we demonstrated that all previous works for optimizing topology in the literature could be incorporated into our theoretical framework. In particular, we obtained convergence results for very general classes of functions with topological flavor computed with persistence theory, and provided corresponding code that one can use to reproduce previously introduced topological optimization tasks. For future work, we are planning to further investigate tasks related to classifier regularization in ML [CNBW19], and to improve on computation time using, e.g., vineyards [CSEM06].

References

[ACG+20] Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda. DTM-based Filtrations. In *Topological Data Analysis*, pages 33–66. Springer, 2020.

[AEK+17] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017.

[AGH+09] Dominique Attali, Marc Glisse, Samuel Hornus, Francis Lazarus, and Dmitriy Morozov. Persistence-sensitive simplification of functions on surfaces in linear time. *TOPOINVIS*, 9:23–24, 2009.

[BCY18] Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and topological inference*, volume 57. Cambridge University Press, 2018.

[BGGSSG20] Rickard Brüel-Gabrielsson, Vignesh Ganapathi-Subramanian, Primoz Skraba, and Leonidas J Guibas. Topology-aware surface

reconstruction for point clouds. In *Computer Graphics Forum*, volume 39, pages 197–207. Wiley Online Library, 2020.

[Bub15] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

[CBO+20] James Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A Zimmer, Julia A Schnabel, and Andrew King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[CCI+20] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796. PMLR, 2020.

[CD18] Frédéric Chazal and Vincent Divol. The density of expected persistence diagrams and its kernel based estimation. In *34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[CdSGO16] Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. SpringerBriefs in Mathematics. Springer, 2016.

[CFL+15] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Subsampling methods for persistent homology. In *International Conference on Machine Learning*, pages 2143–2151, 2015.

[CG20] Gunnar Carlsson and Rickard Brüel Gabrielsson. Topological approaches to deep learning. In *Topological Data Analysis*, pages 119–146. Springer, 2020.

[CNBW19] Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2573–2582, 2019.

[Cos00] Michel Coste. *An introduction to o-minimal geometry*. Istituti editoriali e poligrafici internazionali Pisa, 2000.

[CSEH09] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

[CSEHM10] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have L_p-stable persistence. *Foundations of computational mathematics*, 10(2):127–139, 2010.

[CSEM06] David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In Nina Amenta and Otfried Cheong, editors, *22nd Annual Symposium on Computational Geometry (SoCG 2006)*, pages 119–126. Association for Computing Machinery, 2006.

[DDKL20] Damek Davis, Dmitriy Drusvyatskiy, Sham M. Kakade, and Jason D. Lee. Stochastic subgradient method converges on tame functions. *Found. Comput. Math.*, 20(1):119–154, 2020.

[DL19] Vincent Divol and Théo Lacombe. Understanding the topology and the geometry of the persistence diagram space via optimal partial transport. *arXiv preprint arXiv:1901.03048*, 2019.

[DUC20] Meryll Dindin, Yuhei Umeda, and Frédéric Chazal. Topological data analysis for arrhythmia detection through modular neural networks. In *Canadian Conference on Artificial Intelligence*, pages 177–188. Springer, 2020.

[EH10] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[GC19] Rickard Brüel Gabrielsson and Gunnar Carlsson. Exposition and interpretation of the topology of neural networks. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1069–1076. IEEE, 2019.

[GHO16] Marcio Gameiro, Yasuaki Hiraoka, and Ippei Obayashi. Continuation of point clouds via persistence diagrams. *Physica D: Nonlinear Phenomena*, 334:118–132, 2016.

[GNDS20] Rickard Brüel Gabrielsson, Bradley J Nelson, Anjan Dwaraknath, and Primoz Skraba. A topology layer for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1553–1563, 2020.

[HKND19] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Mandar Dixit. Connectivity-optimized representation learning via persistent homology. In *International Conference on Machine Learning*, pages 2751–2760. PMLR, 2019.

[HKNU17] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems*, pages 1634–1644, 2017.

[KKZ+20] K. Kim, J. Kim, M. Zaheer, J. Kim, F. Chazal, and L. Wasserman. Efficient topological layer based on persistent landscapes. In *NeurIPS 2020 (to appear)*, 2020.

[LOT19] Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. A framework for differential calculus on persistence barcodes. *arXiv preprint arXiv:1910.00960*, 2019.

[MBGY14] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014.

[MHRB19] Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. *arXiv preprint arXiv:1906.00722*, 2019.

[PSO18] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37, pages 13–25. Wiley Online Library, 2018.

[RTB+18] Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. *arXiv preprint arXiv:1812.09764*, 2018.

[SWB20] Elchanan Solomon, Alexander Wagner, and Paul Bendich. A fast and robust method for global topological functional optimization. *arXiv preprint arXiv:2009.08496*, 2020.

[Ume17] Yuhei Umeda. Time series classification via topological data analysis. *Information and Media Technologies*, 12:228–239, 2017.

[Wil96] Alex J Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted pfaffian functions and the exponential function. *Journal of the American Mathematical Society*, 9(4):1051–1094, 1996.

[WLSC20] Fan Wang, Huidong Liu, Dimitris Samaras, and Chao Chen. Topogan: A topology-aware generative adversarial network. In *European Conference on Computer Vision(ECCV)*, 2020.

[ZC05] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.